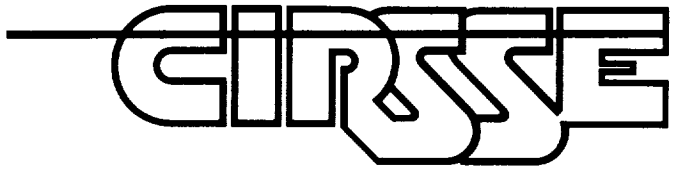

Center for Intelligent Robotic Systems for Space Exploration

Rensselaer Polytechnic Institute
Troy, New York 12180-3590

PLANNING TASKS WITH AN EMERGENT CONNECTIONIST/SYMBOLIC SYSTEM

By:

Michael C. Moed

Center for Intelligent Robotic Systems for Space Exploration (CIRSSE)
Department of Electrical, Computer and Systems Engineering
Rensselaer Polytechnic Institute
Troy, New York 12180-3590

CIRSSE Document #46

PLANNING TASKS WITH AN EMERGENT CONNECTIONIST/SYMBOLIC SYSTEM

A Discussion

Michael C. Moed

Center for Intelligent Robotic Systems for Space Exploration (CIRSSE)
Electrical, Computer and Systems Engineering Department
Rensselaer Polytechnic Institute
Troy, New York 12180-3590

1 Statement of Problem

Rule-based systems for planning abstract robotic tasks often suffer by making rule instantiations which do not achieve the desired rule effects. In many of these instances, general rules "fail" because a specific instantiation is an exception to the general case, and the resulting plan is faulty. In other instances, the specific instantiation fails to achieve the desired effect due to unmodeled perturbations in the environmental state. To overcome these problems, a probability value can be associated with specific instantiations of general rules which quantitatively describes the likelihood that the specific rule achieves the stated effect. From a given initial state, a feasible plan can be developed that satisfies a stated goal by sequencing rules which have highly probable desired effects. The uncertainty that the plan achieves the stated goal can be computed from the probability of effect of each rule used in the plan.

However, since the number of possible instantiations of general rules in a rule store may be excessive, all specific rules cannot be tested thoroughly and maintained in memory with corresponding probability of effect values. Instead, probability of effect values for untested instantiations must somehow be reliably extracted from specific, tested rules.

This work focuses on the problems of: (1) Developing a methodology for finding sets of specific rules which have a high probability of achieving a desired effect from a base which contains many general rules and a limited number of specific instantiations of these general rules, and (2) sequencing rules which have a high probability of effect to develop a plan of abstract tasks which achieves a desired goal.

2 Overview

Consider a planner which attempts to construct a sequence of steps to transform an initial environmental configuration to a goal configuration using rules stored in a rule base. One approach in creating a plan is to find the differences between the goal state and the initial state and determine which rules can reduce this difference by matching the effect of the rule with the goal state. When a

matched rule is applied, the task specified by the rule forms a subgoal. The problem is then reduced to finding a set of rules which can be used to transform the initial configuration to the preconditions of the subgoal. This backward chaining process, called Means-End analysis [NS72] is repeated until a path of subgoal tasks can be found from the goal configuration to the initial state.

Given a rule base of general rules, each rule contains variables which can be instantiated with many object names. At each subgoal step of a backward chaining search, a general rule from the rule base is selected, and the task it specifies forms a subgoal. However, since each general rule may have many different instantiations, this subgoal selection may lead to a large search space. To reduce the size of the space, the search process should select only those instantiations of general rules (called *specific rules*) which have a high probability of causing the desired subgoal effect. To allow this selection to occur, a memory must be maintained which stores the subgoal tasks provided by specific rules along with the probability that the task contained in the rule achieves its stated effect.

Using this memory, each iteration of the search process can recall a set of applicable tasks to be evaluated. An applicable task reduces the difference between the initial and current environmental configurations and is contained in a specific rule which has a high probability of achieving its effect. An uncertainty value can be computed from the probability of effect value of each task's specific rule. One task can be selected from the recalled set to form a plan subgoal by using the uncertainty of the plan as a cost measure to be minimized.

Figure 1 presents a typical tree generated from a search process. In this example, the robot must be holding the bananas to achieve the goal state. This example shows subgoal determination and selection. Each sentence in brackets is a subgoal task selected from the memory and has a high probability of achieving its subgoal effect. Each task is also chosen to reduce the number of differing states between between initial and current object configurations.

3 Identification of Significant Problems

From the above problem scenario, several significant problems can be identified in developing a methodology for planning using rules that have probablistic effects. Given a rule base for manipulating a set of objects in the environment, the problems addressed in this research are:

1. The possible number of general rule instantiations may be extremely large, so it is not possible to test all specific rules and store their probability-of-effect values. Assuming the probability-of-effect value is known for a small number of specific rules (when compared to the total number of possible instantiations), the unknown values for other specific rules must be inferred from known effect probabilities of specific rules. Therefore, a mechanism for extracting similarities and interrelationships between symbols in rules must be developed to determine the probability of effect values of untested specific rules.

2. A method for finding sets of applicable tasks during planning must be developed. An applicable task reduces the current problem by subgoal division and belongs to a specific rule which has a highly probable subgoal effect.

3. Search techniques must be examined which develop low uncertainty task plans. Task plans are formed by orchestrating subgoal determination and expansion using rules with highly probable effects. The uncertainty of a task plan is a function of the probability-of-effect values of each rule in the plan.

To approach a solution to the outlined problems, the following issues are considered:

1. Representations of general rules, specific rules and objects in the environment.

2. An architecture for a memory which distributes the representation of rule tasks and effects so that quantitative interrelationships between symbols can occur.

3. A method for storing specific rule tasks and the probability-of-effect values in the associative memory using quantitative links between symbols. This allows untested specific rules to have their effect probabilities influenced by tested specific rules.

4. A technique for extracting tasks from the associative memory. The recalled tasks belong to specific rules which have a high probability of achieving a particular desired effect. Given the desired effect as input, the associative memory should provide the set of tasks as output. The probability-of-effect value of each rule should also be accessible.

5. A planning technique which accesses the associative memory to develop an ordered set of tasks which form subgoals of a plan. The plan must satisfy a given goal using specific rules with highly probable effects. The planner should attempt to minimize the uncertainty of the plan.

Figure 2 outlines these issues and provides a dataflow description of this work.

4 Method of Approach

4.1 Representation

Before the significant problems can be addressed, a rule representation must be fixed, as well as a representation for objects in the environment. As discussed in [Moe89], a binary representation could be used to represent objects and their states. Each object could be represented by a set of objects states, where a particular state is assigned the value 1 if the object is in that state, 0 if not. In effect, the object and its set of possible states form a schema, as shown in Figure 3. Since a schema must maintain bits for active and inactive object states, the schema size for each object tends to be quite large using this representation. The schema bits for each object are concatenated together to form an environmental state string. This representation suffers because of the large size of the environmental state string.

A higher level representation allows for a more compact description of the environmental state, without loss of functionality. Let us assume that each object in the environment belongs to a particular a priori object class. Each object class is a schema which possesses a set of object state slots that contain the current state of an object that matches the object class. This representation is very similar to object descriptions in such systems as OPS5 [BFKM86]. An example of this representation is shown in Figure 4. Using this representation for objects reduces the size of the environmental state by maintaining only object states which are currently asserted. In effect, this removes the need for 0 bits in the earlier representation. Further, this method for object description allows similar states to be more easily extracted across objects, which is very difficult in the previous representation.

Rules are represented by condition/sentence/effect triples. The sentence portion of a rule is the robotic task which is executed to change the environmental state. Therefore, it is the sentence portion of a specific rule which is used to form the subgoal task during planning.

In [Moe89], the condition and effect portions of a rule were full environmental state strings. This led to a sizable representation for a rule. Also, using a full environmental state for condition and effect turned the rule selection process into a database lookup, since a rule could not exist in the rule store unless it had been experimented with. This led to a huge storage requirement for the set of rules.

To reduce the size of each rule and the size of the rule store, the following framework is adopted. Let the rule store maintain two types of rules, general and specific. Both types of rules are condition/sentence/effect triples. The form of the general rules are as follows:

General Rules:

1. Condition: Pairs of object class variables and object state variables. The object class variable may already be instantiated with a particular object name. The object state variable may already be instantiated with a particular object state. The object state variable is dependent on the schema slot types allowable with the given object class.

2. Sentence: Actor, Action, Direct Object and Indirect Object quadruple where Actor, Direct Object, and Indirect Object are object class variables. An object class variable in the sentence may be instantiated with a particular object name. The Action is one of a set of possible action names.

3. Effect: Pairs of object class variables and object state variables. The object class variable may already be instantiated with a particular object name. The object state variables may contain a modified value reflecting the effect of the sentence on the environment.

An example of general rules is presented in Figure 5. As shown, general rules are similar to uninstantiated rules in some expert systems.

Specific rules are general rules with all object class name variables instantiated with object names, and all the schema slots instantiated with object states. Specific rules also maintain a probability-of-effect value, which is the likelihood that the rule sentence will cause the rule effect given the rule conditions. The form of specific rules is as follows:

Specific Rules

1. Condition: A set of object name and object state pairs. The state values fill schema slots.

2. Sentence: Actor, Action, Direct Object and Indirect Object quadruple where Actor, Direct Object and Indirect Object are object names and the Action is one of a set of possible action names.

3. Effect: A set of object name and object state pairs. The object states are modified schema slot values representing the change in the environment due to the execution of the rule sentence.

4. Probability-of-effect: An experimentally determined value between 0.0 and 1.0 reflecting the likelihood of the specific effect occuring.

An example of specific rules is presented in Figure 6. Both the general rule base and a set of specific rules are provided to the planning system from an external mechanism. While it is easier to create the general rules in a top-down fashion, specific rules can be developed bottom-up from general rules through an external algorithm such as PLAY [Moe89]. As shown, a particular condition/sentence pair can have multiple effects whose probabilities sum to 1. The probability-of-effect value of specific rules provides a performance assessment of the applicability of general rules in different situations.

This representation of rules and objects is suitable for planning abstract tasks similar to those planned by domain-independent planners such as STRIPS [FN71]. It can be directly applied to the Organization Level of the Intelligent Machine [Sar79] as described in [Moe89], which requires abstact reasoning as part of its hierarchical intelligence system. It is important to realize that this type of representation is not well suited for tasks which require more detailed descriptions of objects, such as geometric based planning.

4.2 Associative Rule Memory

An associative rule memory must be developed which can:

1. Distribute the representation of rule sentences (tasks) and effects over individual symbols.

2. Allow quantitative interrelationships between symbols.

3. Store rule sentences and corresponding probability-of-effect values in the quantitative links between symbols.

4. Develop probability-of-effect values for untested specific rules based on tested specific rules which have been stored in the associative memory.

5. Allow associative recall of rule sentences and probability-of-effect values given a desired effect as input.

Connectionist networks (or artificial neural networks) provide a suitable architecture for representing interrelationships between symbols and for achieving associative recall. One connectionist framework appropriate for this type of problem is the Boltzmann Machine [HS86]. In this framework, nodes are used to represent symbols, and weights (or connections) between nodes maintain cross-correlation information about pairs of nodes. The goal of the network is to find a valid set of asserted nodes which are highly correlated. In other words, the weights form a set of weak constraints between nodes, and satisfaction of the weak constraints by node assertion corresponds to the associative recall, or output of the network for a given input.

Several related topics have been examined using Boltzmann Machine architectures. One recent system uses a Boltzmann Machine for Schema recall [RSMH86]. In this work, link weights are a function of symbol co-occurance using a one node per symbol mapping. Schema recall is achieved through local minimization of the Energy function using an asynchronous node search technique developed by Hopfield [Hop82]. Another effort has used Boltzmann Machines to represent a small production rule system [TH85]. Using about 8000 nodes, the six rule production system can do single variable instantiation in a selected rule, and chain the results of one rule to fire another. A distributed representation was used over the nodes to create the rules and working memory elements. The weights in the network were fixed once assigned, so no new rules or working memory values could be modified or added.

In the system proposed by this paper, the nodes of the Boltzmann Machine represent the symbols of both the sentence and effect of a specific rule. Distributing sentences over individual symbols allows sentences with subsets of identical symbols to share the same nodes and connection weights in the Boltzmann Machine. By training the weights of the Boltzmann Machine, one can allow sentence subsets which lead to more probable effects to influence sentences of untested specific rules sharing the same nodes and connections. Similarly, sentence subsets leading to less probable effects can influence nodes and weights of untested specific rule sentences and effects. Since the nodes represent symbols, the weights can be used to determine which sets of symbols in a sentence should be active for a desired effect input by searching for the maximum correlation between active nodes in the network.

There are several research issues involved with the use of Boltzmann Machines. First, one must determine a suitable set of nodes and a connectivity pattern between the nodes for the required set of recall tasks. This defines the architecture of the Boltzmann Machine. Second, a learning algorithm must be developed for assigning the weight values on the connections which in turn shapes the landscape of effect probabilities. Finally, search algorithms must be investigated which find node configurations which correspond to tasks of specific rules which have a high probability of achieving a given effect.

4.2.1 Architecture of the Boltzmann Machine

Figure 7 presents a Boltzmann Machine architecture for the associative memory. Each node level represents a set of symbols in a rule. From top to bottom, the labeled levels are:

1. N: Object Name in Desired Effect

2. S: State of Object in Desired Effect

3. A: Actor of Sentence

4. V: Action of Sentence

5. D: Direct Object of Sentence

6. I: Indirect Object of Sentence

Each node on each labeled level represents one possible instantiation of all symbols in that symbol class. For example, the Actor node level may contain 3 nodes corresponding to the actors ARM1, ARM2 or ROBOT. Similary, the Action node level contains individual nodes for GRASP, RELEASE, MOVE-TO, etc. Unlabeled levels contain *hidden nodes* which represent pairwise combinations of sentence symbols.

Connections are made between nodes of different levels. Each connection contains a modifiable weight which represents the constraint that one node places upon another. Since only one Actor, one Action, zero or one Direct Objects and zero or one Indirect Objects may be part of any sentence, only one node on each of these levels may be active at a time. Therefore, there is no need for connections between nodes of the same level for these symbol classes. Similarly, only one Object Name node and one Object State node are asserted for each Desired Effect, so connections are not required between nodes of the same level for these classes.

The connection weights between nodes are used to store the probability-of-effect values for the specific rules represented in the network by asserted sentence and effect nodes. These probability values are derived from a "goodness" measure called Energy, which represents the amount of correlation between a set of asserted nodes. A set of asserted nodes is called the *configuration* of the network, and for a particular set of asserted nodes $n = (n_0, n_1, \cdots, n_k)$, the Energy of the configuration is given by:

$$E(n) = \frac{1}{2} \sum_i \sum_j w_{ij} n_i n_j \tag{1}$$

where w_{ij} is the weight between nodes n_i and n_j and all n_i, $n_j \in n$. The system is defined such that lower Energy network configurations represent more highly correlated asserted nodes. Highly correlated nodes form configurations which have high probability-of-effect values.

The term "Boltzmann Machine" is derived from the manner in which the Energy of a configuration is related to the probability of the configuration. Using a Boltzmann distribution analog, the relative probability between two network configuations is given by:

$$\frac{P(n^\alpha)}{P(n^\beta)} = e^{-(E(n^\alpha) - E(n^\beta))} \tag{2}$$

where n^α, n^β are two network configurations, and $P(\cdot)$ is the probability of the network being in that configuration. Since the configuration probability represents the probability that a particular sentence and effect are asserted on the network nodes, this value can be used as the probability-of-effect value for specific rules. Therefore, the probability-of-effect value for specific rules can be represented in the connection weights between nodes in the Boltzmann Machine.

Associative recall is accomplished by asserting the Object Name node and Object State node in the Desired Effect levels. These nodes form the input to the network and are fixed during recall. The network then searches for a set of sentences which produce the desired effect by finding sets of asserted nodes on the sentence node levels which place the network in a low Energy configuration. Low energy network configurations correspond to sentences which have a high probability of achieving the desired effect. Concatenation of asserted nodes representing Actor, Action, Direct and Indirect Objects forms the sentence produced by associative recall.

This representation was chosen because it allows a sentence to be distributed over a set of separate symbols, and can allow particular symbols to achieve correlational relationships through weight adaptation in training. Also, the relationship between symbols and desired effects is distributed across the representation. These relationships allow symbols in untested specific rules to have some probability-of-effect value assigned based on a comparitively small base of tested rules which share symbols and connections.

4.2.2 Training the Boltzmann Machine

To store specific sentences and effects in the associative memory, the Energy landscape must be modified so sentences with a high probability of achieving particular effects correspond to low Energy configurations of the network. To accomplish the Energy landscape modification, the weights between nodes of the network have to be adjusted. While techniques such as Error Backpropagation [RM86] are used to adjust weights for feedforward pattern classification tasks, they do not apply well to associative retrieval based on Energy minimization. A Boltzmann Machine learning technique has been developed for pattern classification, but it is quite unwieldy for large network sizes [AHS85]. Due to the structure of the problem at hand, it may be possible to modify this technique and make it admissible for probabilistic rule representation. Techniques such as those presented in [RSMH86] for schema retrieval are based on the frequency of occurance of pairwise nodes, and provide another direction to investigate for training the associative memory.

A training procedure generally consists of asserting a sentence and its effect symbols on the corresponding nodes of the Boltzmann Machine. The connection weights between asserted nodes are adjusted according to the probability-of-effect value stored in the specific rule. If the probability is low, the weights are increased. If the probability is high, the weights are decreased in order to reduce the Energy value for this configuration. The exact methodology for modifying the weights is a research task that this work will address.

Also addressed will be a method for training the network on the general rules in the rule store. Training the network on a general rule creates a set of low Energy configurations for all possible instantiations of the general rule. These configurations are then modified when the network is trained on specific rules.

4.2.3 Search Techniques for the Boltzmann Machine

One of the significant problems addressed in this research is finding sets of specific rules which have a high probability of achieving a desired effect. In the previous section, it was shown how a Boltzmann Machine architecture could be used to store sentences, effects, and probability-of-effect values of specific rules as a set of nodes and weights. Given a desired effect, the Energy of the network can be minimized, yielding a network configuration which is the best sentence for achieving the effect. Using this Energy value, the probability-of-effect value can be calculated. This section describes search techniques which can be used to find the low Energy sentences corresponding to sets of specific rules with highly probable effects that can be used as subgoals in planning.

Sentence search methods for this connectionist architecture must subsume the following criteria:

1. The Energy of a network configuration is used as the cost function to be minimized by the search.
2. Each element generated by the search (search element) must be mapped to a binary representation corresponding to asserted sentence nodes in the network.
3. A search element is valid if it maps to one node per sentence node level.
4. The "goodness" of a valid search element is proportional to the network Energy once the element is mapped and asserted on the sentence nodes. Low Energy configurations reflect better search elements.

5. The search technique must find several low Energy sentences.

6. The search technique must be known to converge to the minimum Energy of the network, when allowed.

Experimentation has been done on search techniques which can be adapted to meet the above criteria. The experiments conducted evaluated the time to reach the minimum Energy configurations of a fifteen node Boltzmann machine, where each node was connected to every other node. The search techniques evaluated were Simulated Annealing, Modified Genetic Algorithm, and Random Search. The Modified Genetic Algorithm is a form of the Genetic Algorithm [Hol75]. The Modified version has been proven to converge to the minimum of a cost function, and has achieved good performance in the conducted experiments [Moe89, MS89, SM88].

Further research must be done to develop methods for using the algorithms to find multiple low Energy sentences instead of a single solution. Also, since the number of valid search elements is a small subset of the possibly generated search set, a methodology must be developed to force the search algorithm to generate only valid elements.

4.3 Planning

Using a Boltzmann Machine architecture, the associative memory extracts similarities between symbols in rules and provides probability-of-effect values for untested specific rules. The associative memory can find sets of specific rules with highly probable effects and which divide a problem into subgoals using search techniques that meet the criteria described above. Planning consists of determining which subgoal specific rules generated by the Boltzmann Machine should be expanded and which paths should be explored.

The search technique employed for subgoal determination in planning is Means-End Analysis. As discussed above, this technique attempts to reduce the difference between the initial configuration of objects and a desired end configuration by a backward chaining search. Subgoal selection is based on minimizing a cost measure. The cost measure to be minimized in this system is the uncertainty of the plan, and is an entropy function of the probability-of-effect values of rules used in the plan [Moe89]. When used with the associative memory, the recursive algorithm proceeds as follows:

1. Construct the set of object state differences between the initial configuration and the current desired configuration.

2. For each object state difference:

   (a) This difference is a desired effect. Assert the object name and desired state on the corresponding nodes in the Boltzmann Machine.

   (b) The Machine will produce a set of applicable sentences which have a high probability of achieving this effect. The number of sentences in the set is limited to a certain maximum, and each sentence must have an effect probability higher than an a priori threshold.

   (c) Compute the uncertainty of each applicable sentence.

   (d) Select a sentence from the applicable set based on plan cost. This is a subgoal task.

   (e) Determine the cost of the path in the plan as a function of the uncertainty of this sentence and decide whether to pursue.

   (f) If too expensive, examine other sentences or return to previous decision level.

   (g) If not too expensive, find the sentence preconditions in the rule store and instantiate sentence variables and the initial state of the environment to the preconditions. If all preconditions are satisfied, this subgoal has been satisfied. Return this solution.

(h) The set of unsatisfied preconditions now form subgoals and represent the current desired configuration. Recur to 1.

3. If all subgoals are satisfied, a plan has been found, else report failure.

Referring again to Figure 1, this algorithm forms an AND-OR graph for planning where AND arcs represent multiple object state changes which must occur to satisfy the goal state, and OR arcs represent a choice among multple low uncertainty sentences which can be used in developing the plan. A best first search can be used on this graph for determining which OR paths to expand based on cost.

Cost metrics must be developed for this search. Possible metrics are total uncertainty of a plan, or maximum uncertainty of a rule in a plan. Included in the metric may be a measure of the number of sentences in the plan in an attempt to trade off uncertainty with plan length.

Planning robotic tasks using these techniques may required additional mechanisms such as constraint passing, which is included in such systems as MOLGEN [Ste81a, Ste81b]. Constraints occur in planning when a particular object state is assumed by a planning step, and cannot be altered. Since the Boltzmann Machine is used to generate plan subgoals, constraints can be used to alter the Energy values for unusable sentences by biasing particular node combinations. Constraints can also be used to guide the selection of paths for exploration in the AND-OR graph.

5 Evaluation

The following items will be used to evaluate the work and determine its completion.

- Training the Boltzmann Machine:
  - Demonstrate the effectiveness of developed weight adaptation algorithms on the Energy landscape of the Boltzmann Machine.
  - Construct a set of specific rules with probability-of-effect values. Train the network on the set of specific rules using a developed weight adaptation algorithm. After training, assert the sentences on the nodes of the Boltzmann Machine and determine if the probability-of-effect values are consistent.
  - Construct a set of untested specific rules. Heuristically, assign a range of expected probability-of-effect values for each untested rule. Assert the sentences of these rules on the nodes of the previously trained Boltzmann Machine. Determine if the probability-of-effect values are consistent with the heuristic values and evaluate results if different.
- Searching for sentences of specific rules with highly probable effects in the Boltzmann Machine.
  - Construct a network with known Energy minima. Test search techniques on this network using search time and minima discovery as the success criteria.
  - Evaluate search techniques on the Boltzmann Machine constructed through the above training set by searching for sets of specific rules with highly probable effects.
- Planning
  - Devise a case study and show that the planning algorithm can successfully construct a set of sentences which achieve a goal and minimize the uncertainty cost criteria.
  - Construct a set of constraints for the case study and demonstrate the ability to propagate constraints through alteration of the Energy landscape and limited subgoal selection.

6 Summary

This paper presents a planning system which uses an associative memory to select sets of tasks for transforming a set of objects from an initial state to a final configuration. The tasks are containted in rules which have probabilistic effects. The planning system attempts to minimize the uncertainty of a plan by selecting only those tasks which belong to rules which have highly probably effects.

This work assumes that the following structures are provided:

- A set of object class schemas.
- A list of objects.
- A rule base with general rules.
- A set of specific rules with probability-of-effect values.

The significant problems addressed in this work are:

- The development of a mechanism for extracting similarities or interrelationships between symbols in rules to provide probability-of-effect values for untested specific rules.
- A method for finding sets of tasks which develop planning subgoals. These tasks are sentences contained in specific rules which have highly probable effects.
- The development of a search technique which creates low uncertainty task plans by expanding subgoals and exploring low cost paths.

The further research required to address these problems is:

- Development of training techniques for altering the sentence Energy values by modifying connection weights.
- A methodology for training the Boltzmann Machine on general rules.
- Development of search methods for finding sets of low Energy sentences in the Boltzmann Machine.
- Implementation of a planning algorithm using a best first search of an AND-OR graph with possible constraints.
- Determination of a set of cost metrics for selecting subgoal sentences.
- Development of a case study to test and evaluate the system using a generated set of general rules and a set of specific rules for a set of tasks in a robotic environment. This case study should be relevant to the Organization level of the Intelligent Machine.

References

[AHS85] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski. A learning algorithm for Boltzmann machines. *Cognitive Science*, 9:147–169, 1985.

[BFKM86] L. Brownston, R. Farrell, E. Kant, and N. Martin. *Programming Expert Systems in OPS5*. Addison-Wesley, 1986.

[FN71] R. E. Fikes and N. J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3,4), 1971.

[Hol75] J. H. Holland. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, Ann Arbor, MI, 1975.

[Hop82] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79:2554–2558, 1982.

[HS86] G. E. Hinton and T. J. Sejnowski. Learning and relearning in Boltzmann machines. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing Volume I.* The MIT Press, Cambridge, MA, 1986.

[Moe89] M. C. Moed. The Organizer: Planning tasks with an emergent connectionist/ symbolic system. Technical Report CIRSSE-42, Center for Intelligent Robotic Systems for Space Exploration (CIRSSE), Rensselaer Polytechnic Institute, Troy, NY 12180-3590, October 1989.

[MS89] M. C. Moed and G. N. Saridis. A Boltzmann machine for the organization of intelligent machines. In *Proceedings of the NASA Conference on Telerobotics*, Pasadena, CA, January 1989.

[NS72] A. Newell and H. A. Simon. *Human Problem Solving.* Prentice-Hall, 1972.

[RM86] D. E. Rumelhart and J. L. McClelland. *Parallel Distributed Processing, Vol. I.* The MIT Press, Cambridge, MA, 1986.

[RSMH86] D. E. Rumelhart, P. Smolensky, J. L. McClelland, and G. E. Hinton. Schemata and sequential thought processes in PDP models. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing Volume II*, pages 7–57. The MIT Press, Cambridge, MA, 1986.

[Sar79] G. N. Saridis. Toward the realization of intelligent controls. *IEEE Proceedings*, 67(8), 1979.

[SM88] G. N. Saridis and M. C. Moed. Analytic formulation of intelligent machines as neural nets. In *Proceedings of the IEEE Conference on Intelligent Control*, Washington, DC, August 1988.

[Ste81a] M. Stefik. Planning with constraints (MOLGEN: part 1). *Artificial Intelligence*, 16, 1981.

[Ste81b] M. Stefik. Planning with constraints (MOLGEN: part 2). *Artificial Intelligence*, 16, 1981.

[TH85] D. S. Touretzky and G. E. Hinton. Symbols among the neurons: Details of a connectionist inference architecture. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 239–243, Los Angeles, CA, 1985.

TYPICAL PLAN TREE

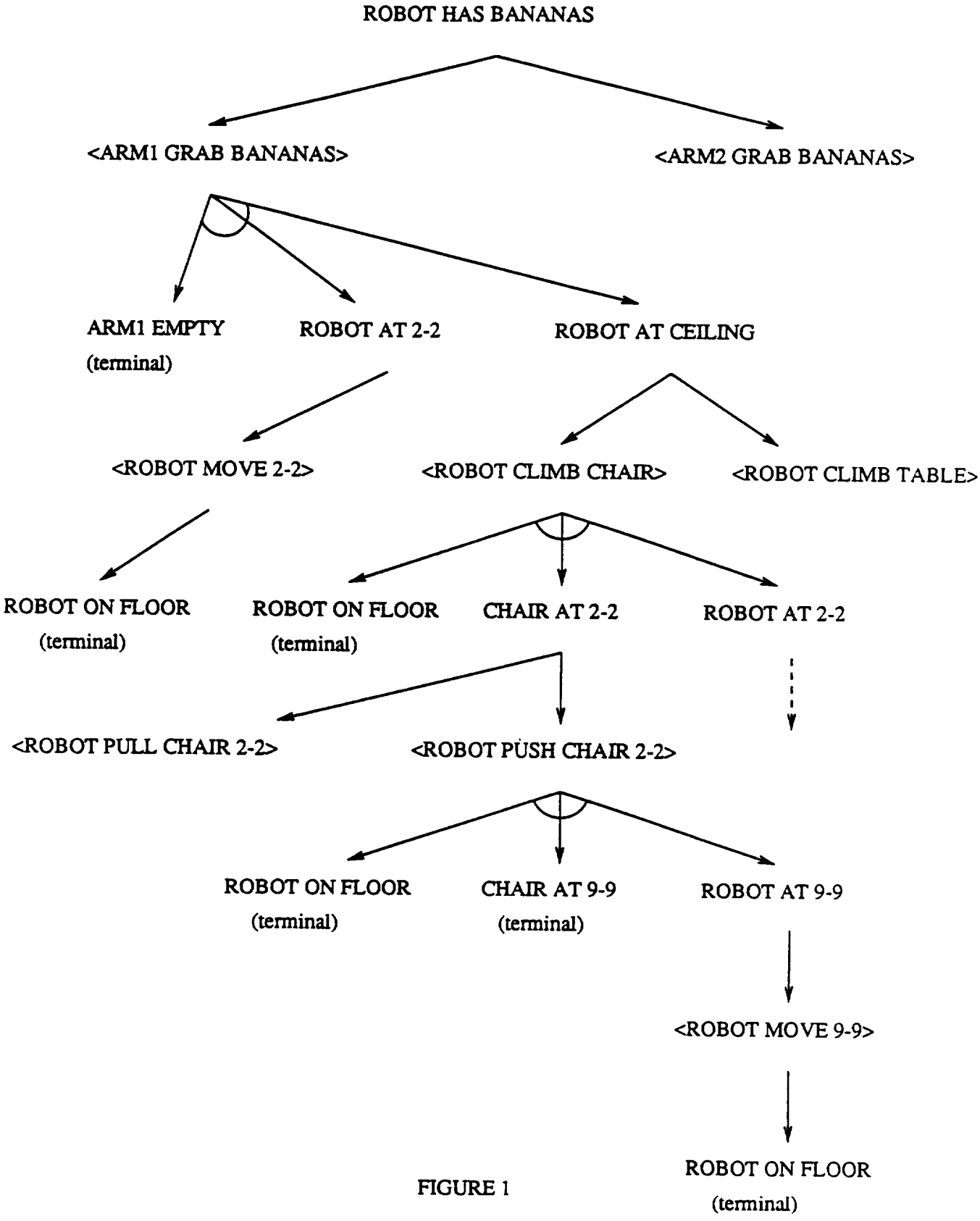

FIGURE 1

PLANNING SYSTEM BLOCK DIAGRAM

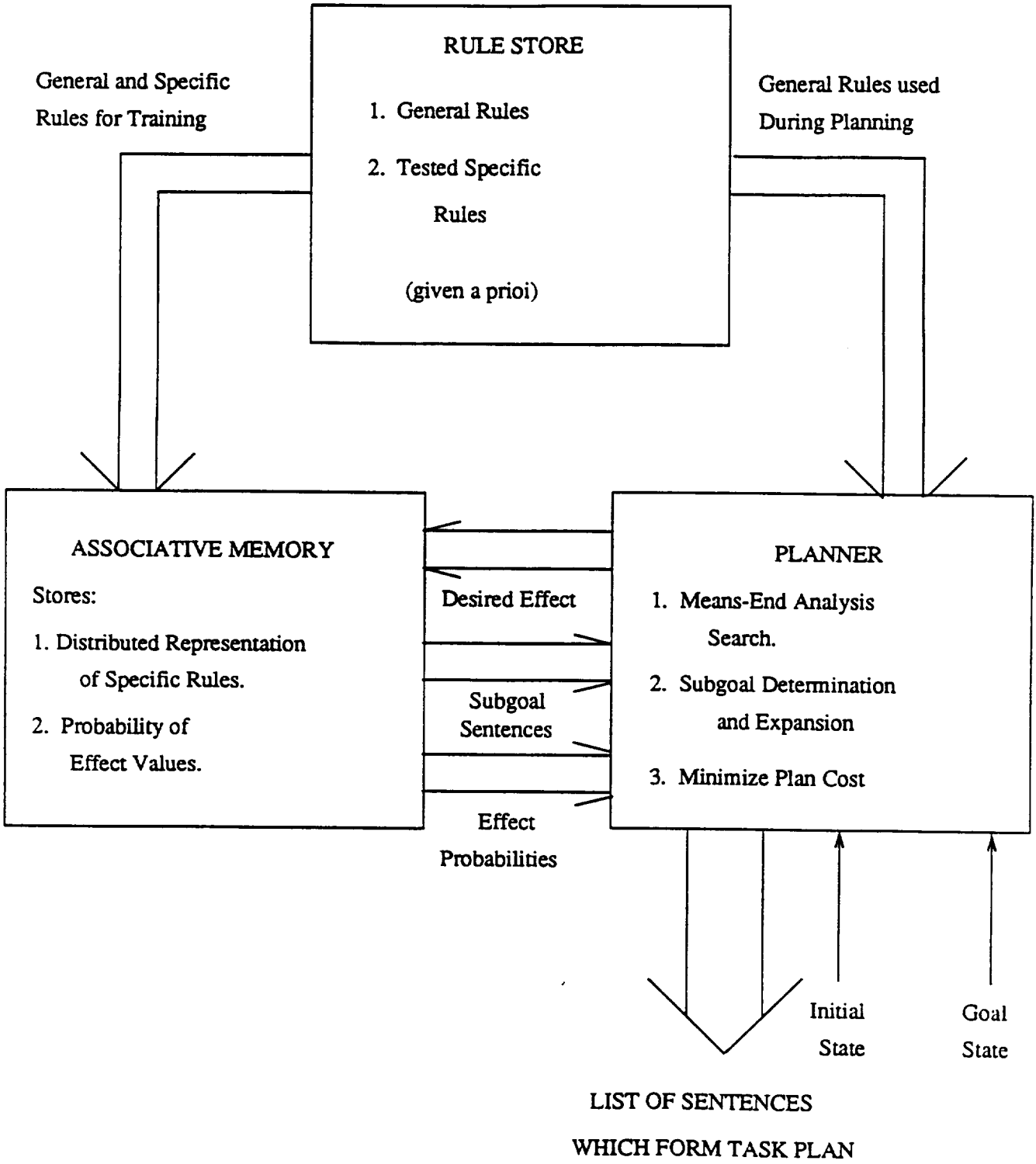

FIGURE 2

Binary Schema Example

Object: Wrench

State	Value
Wrench on Table	0
Wrench on Floor	0
Wrench in Box	0
Wrench in Arm1	1
Wrench in Arm2	0
Wrench Open	0
Wrench Closed	1

FIGURE 3

Object Class Schema Examples

Object Class:	tooltype
Name:	Wrench
Location:	Table
State:	Open

Object Class:	manipulatortype
Name:	Arm1
Location:	2-2
Possession:	Plyers

FIGURE 4

GENERAL RULE EXAMPLE

Conditions:

*manipulatortype*1 location x
*objecttype*1 location x height y
Robot height y

Sentence:

*manipulatortype*1 grasp *objecttype*1

Effect:

*manipulatortype*1 possession *objecttype*1

FIGURE 5

SPECIFIC RULE EXAMPLE

Conditions:

Arm1 location 2-2
Bananas location 2-2
Robot height Ceiling

Sentence:

Arm1 grasp Bananas

Effects:

Arm1 possession Bananas
Probability: 0.98
Bananas height Floor
Probability: 0.02

FIGURE 6

NETWORK DIAGRAM AND PARTIAL WEIGHT DESCRIPTION

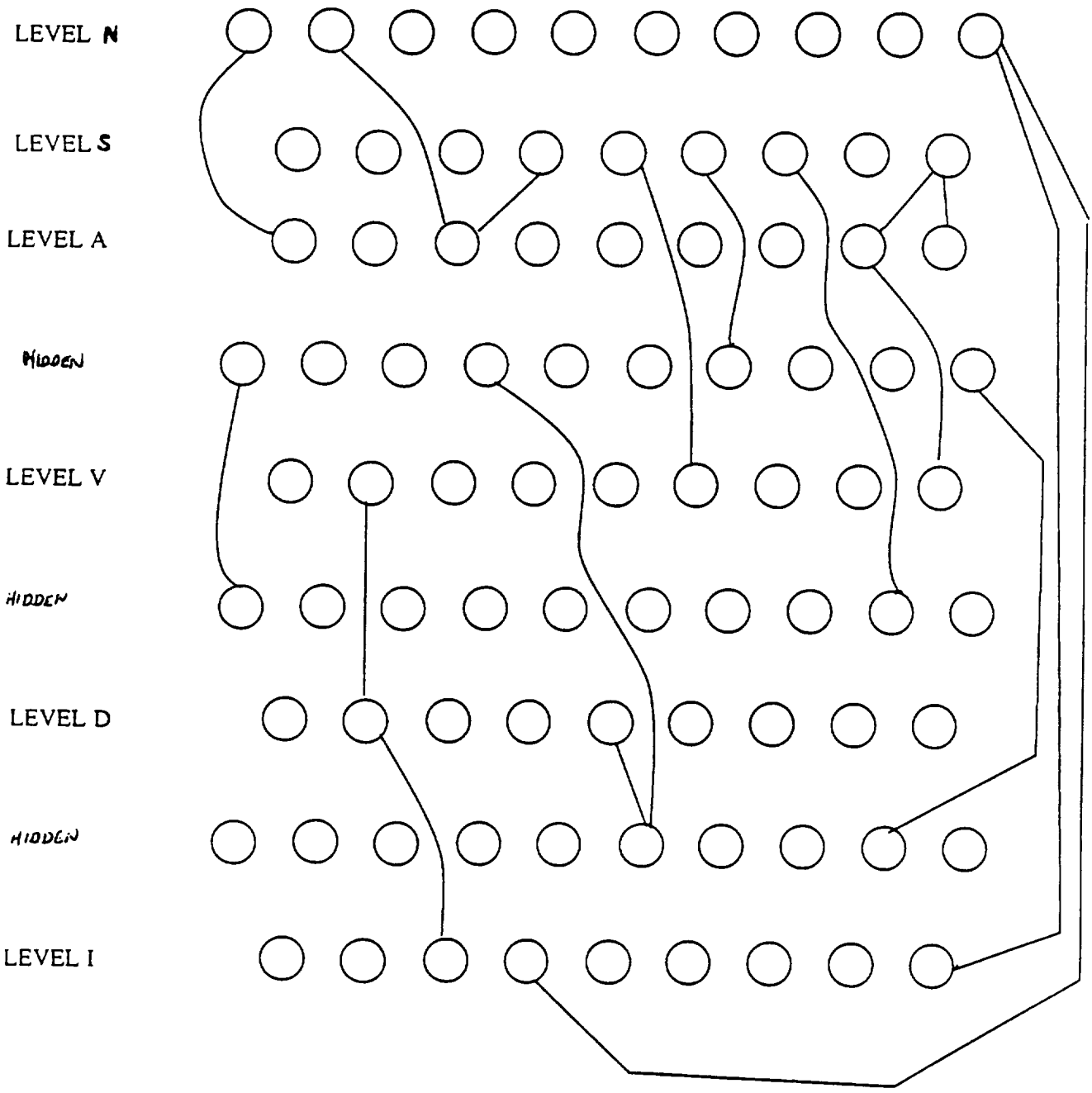

Figure 7